DA53: Compilation and Language Theory - Mid-Term Exam A2024

Duration: 1h30. No document allowed.
English recommended, French accepted.

Part 1: Introduction (4 points)

Question 1.1:

What is a parse tree in the context of a compiler? Describe the the content of a parse tree and one of its possible usage.

Question 1.2:

What is a Deterministic Finite Automata in the context of a compiler? And, what is the usage of a Deterministic Finite Automata in a compiler?

Question 1.3:

List the different stages that are followed by a compiler for generating binary executable code from a source program.

Part 2: Lexical Analysis (7 points)

Let the language that is composed of:

1. the number constants (integer or decimal number, without the exponential part), e.g., “1.456”.
2. the two comparison operators “>”, “==” that compare two operands.
3. the two arithmetic operators “+”, “*” that are computing an addition and a multiplication respectively.
4. arithmetic and comparison expressions may be written with parentheses, e.g. “(1+5)*5”.
5. the “print v” statement that print out its parameter “v”. “v” is an expression compose of constants, comparison or arithmetic operators. There is no variable in this language.

6. the "if-then-else" statement: `if CONDITION then STATEMENT else STATEMENT`, where `CONDITION` is composed of numeric constants and comparison operator and `STATEMENT` may be another "if-then-else" statement or a "print v". The "else" statement is not mandatory, such that it is possible to write `if CONDITION then STATEMENT`.

Example:

```
if 4 > 5 then
  if 7 == (4 + 5.25) then
    print 1
  else
    print 2 * 1.54
```

Question 2.1:

What is the alphabet of this language?

Question 2.2:

Write the table that is matching the regular expressions, the lexemes, the tokens and the attributes of the tokens in four columns.

Token name	Example(s) of lexemes	Regular expression	Attribute if any

Question 2.3:

Draw the Nondeterministic Finite Automata (NFA) that is recognizing the language tokens. In a NFA, the nodes represents the current state (or step) in the recognition of a token. The edges in the NFA have a label with one or more characters that represent the character(s) to be recognized from a given state of the NFA. The label of a NFA edge may be ϵ for representing a transition with "nothing".

Part 3: Syntax Analysis (6 points)

In this part, you must use the same language as in Part 2.

Question 3.1:

Write the grammar rules (using the Backus Naur Form - BNF) for the language that is described in Part 2. You grammar will be analyzed with LL(0) approach. So that, you should avoid left-recursions and remove ambiguities from the grammar productions.
Reminder: A BNF grammar is composed of productions that have a head on the left, and a sequence of terminals (tokens) and nonterminals on the right.

`NONTERMINAL → terminal <NONTERMINAL> terminal <NONTERMINAL>`